VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozpoznání názvů ulic z obrazů

Street Names Recognition from Images

2019 Ondřej Kössler

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Kössler**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Rozpoznání názvů ulic z obrazů
Street Names Recognition from Images

Jazyk vypracování: čeština

Zásady pro vypracování:

Detekce názvů ulic je zajímavým problémem v analýze obrazu. Cílem práce je ověření kvality detekce a rozponání dostupných modelů pro strojové učení na českých názvech ulic, případně jejich vylepšení. K tomuto využijte dostupných vývojových frameworků pro strojové učení.

Ve své práci proveďte:
1. Nalezněte a prostudujte dostupné modely strojového učení, které jsou schopny detekovat názvy ulic, případně jiné informační cedule.
2. Proveďte otestování takovýchto modelů na vhodně zvoleném datasetu, který obsahuje i české názvy (případně takovýto dataset sestavte).
3. Pokuste se případně upravit model tak, aby lépe rozpoznával české názvy.
4. Dosažené výsledky náležitě zhodnoťte.

Seznam doporučené odborné literatury:

[1] StreetView Tensorflow Recurrent End-to-End Transcription (STREET) Model, https://github.com/tensorflow/models/tree/master/research/street
[2] Attention-based Extraction of Structured Information from Street View Imagery, https://github.com/tensorflow/models/tree/master/research/attention_ocr

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura, Ph.D.**

Datum zadání: 01.09.2018
Datum odevzdání: 12.07.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. června 2019 .....

Rád bych poděkoval především vedoucímu bakalářské práce Ing. Jan Gaura, Ph.D. za neocenitelnou pomoc po odborné a technické stránce při konzultacích.

Abstrakt

Účelem této bakalářské práce je prostudovat a ověřit kvalitu detekce dostupných modelů pro strojové učení schopných rozpoznat název ulice a jiné informační cedule. Poté podobný model navrhnout, implementovat a otestovat na obrazech ve vysokém rozlišení obsahujících české názvy.

Klíčová slova: detekce textu; rozpoznání textu; název ulice

Abstract

The purpose of this bachelor thesis is to study and verify the quality of detection of available models for machine learning which are capable to recognize street name and other information signs. Then design, implement and test a similar model on high-resolution images containing Czech names.

Keywords: Text Detection; Text Recognition; Tensorflow.js; Street Sign

Obsah

Seznam použitých zkratek a symbolů

API	–	aplikační programové rozhraní
GPU	–	grafická procesorová jednotka
CPU	–	centrální procesorová jednotka
OCR	–	optické rozpoznávání znaků
JS	–	JavaScript - programovací jazyk

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Bakalářská práce má za cíl prozkoumat dostupné modely pro detekci názvů ulic, případně i jiných textů z obrazů, otestovat takové modely na datasetu obsahujícím i české názvy ulic a navrhnout vlastní řešení, které dokáže přečíst název ulice z obrazu. V teoretické části se seznámíme se základními pojmy z oblasti analýzy obrazů, strojového učení a použitými technologiemi při implementaci vlastního postupu. V kapitole věnované analýze bude představeno několik prací zabývajících se řešením podobných problémů a rozebrání jejich postupů, dále pak také představení vlastního postupu při řešení problému. Praktická část práce bude věnovaná prezentaci výsledků dosažených při testování sestaveného datasetu obrazů s českými názvy ulic s použitím již existujícího řešení a vlastního postupu.

2 Teorie

Tato kapitola slouží k seznámení a definování jednotlivých pojmů z oblasti zpracování obrazů a neuronových sítí, které budou využívány v této bakalářské práci. Kromě toho také představením využitých knihoven a metod včetně případné ukázky použití.

2.1 Digitální zpracování obrazu

Jedná se o manipulaci s digitálními obrazy pomocí počítačů. Na vstupu je obraz pořízený kamerou, na který je aplikována úprava pomocí různých algoritmů jako je detekce hran, segmentace obrazu, změna rozlišení nebo odstranění šumu, rozklad obrazu, korespondence dvou obrazů, detekce geometrických primitiv (přímek, rohů, kružnic, elips). Množství technik pro zpracování obrazu bylo vyvinuto okolo roku 1960 v Jet Propulsion Laboratory pro zpracování obrazů pořízenými vesmírnými sondami [1]. Velké využití zpracování obrazu je v oborech jako je robotika, lékařská diagnostika (např. zpracování obrazů z ultrazvuku, magnetické rezonance, CT mozku), ale i v průmyslu (rozpoznávání otisků prstů či sítnice, záznamů bezpečnostních a dopravních kamer, apod.).

2.2 Počítačové vidění

Počítačové vidění má dva hlavní cíle. Z biologického pohledu se snaží napodobit lidské schopnosti vidění a rozpoznávání objektů a z technického hlediska má za cíl vyvinout autonomní systémy, které by mohly plnit podobné, popř. i složitější úkoly než lidský vizuální systém. Oba pohledy jsou samozřejmě úzce spjaty. Vlastnosti lidského vizuálního systému často dávají inspiraci pro inženýry, kteří navrhují systémy počítačového vidění. Naopak algoritmy počítačového vidění mohou nabídnout pohled na to, jak funguje lidský vizuální systém. Počítačové vidění jako oblast výzkumu je velmi obtížná. Téměř žádný zkoumaný problém nebyl uspokojivě vyřešen. Jedním z hlavních důvodů tohoto problému je, že lidský vizuální systém je příliš dobrý pro úkoly jako např. rozpoznávání obličeje, počítačové systémy nedosahují takové úspěšnosti, jaké je schopen člověk, který dokáže rozpoznat tváře pod všemi druhy osvětlení a různě deformovanými scénami. Také se zdá, že neexistuje žádný limit na počet tváří, které si člověk dokáže zapamatovat. Nedaří se vyvinout systém který by předčil člověka v těchto ohledech. Mezi současné aplikace počítačového vidění patří například autonomní navigace aut [2].

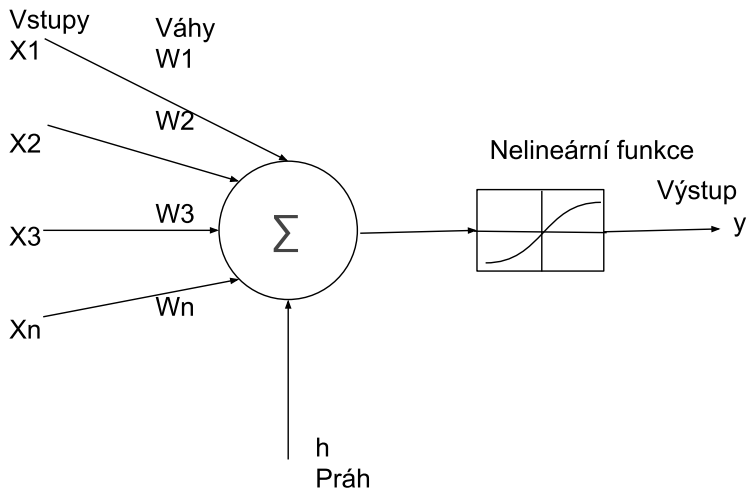
2.3 Umělá inteligence a neuronové sítě

V informatice pojem umělá inteligence (AI), značí inteligenci užívanou stroji. Na rozdíl od přirozené inteligence, kterou známe u lidí a zvířat. Počítačová věda definuje výzkum AI jako studium „inteligentních agentů“ - jakéhokoliv zařízení, které vnímá své prostředí a podniká kroky, které maximalizují jeho šanci na úspěšné dosažení svých cílů [3]. Umělá neuronová síť je založena na množině spojených jednotek zvaných umělé neurony, jenž napodobují neurony

v biologickém mozku. Každé spojení, podobně jako synapse v mozku, může přenášet signál z jednoho neuronu do druhého. Umělý neuron, přijímá signál, který zpracuje a následně může poslat signál dalším umělým neuronům, které jsou k němu připojeny. Signály se transformují pomocí určitých přenosových funkcí (1). Takový neuron má libovolný počet vstupů, ale pouze jeden výstup, jako na obrázku 1. Existuje mnoho modelů neuronů. Od velmi jednoduchých, které používají nespojité přenosové funkce, až po velmi složité, jenž popisují přesně neuron živého organismu. Model neuronové sítě se pak skládá z několika vzájemně propojených neuronů. Matematicky můžeme funkci umělého neuronu popsat následovně:

$$y = F\left(\sum_{i=1}^{n} X_i W_i + \Theta\right), \tag{1}$$

kde X_i je hodnota na i-tém vstupu, W_i je váha i-tého vstupu, Θ je prahová hodnota, n je celkový počet vstupů, F je obecná ne-lineární funkce a y je hodnota výstupu.



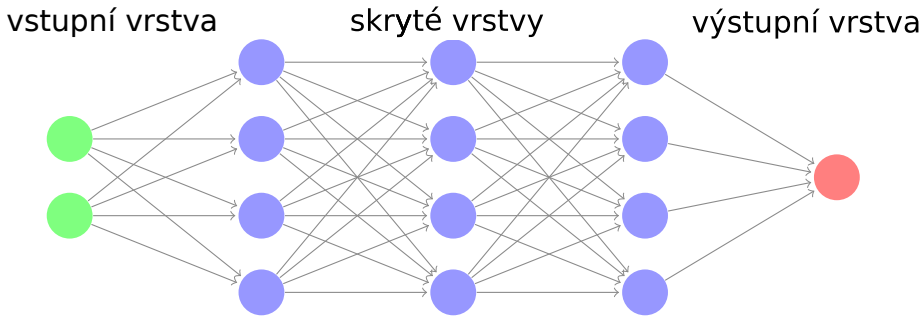
Obrázek 1: Model umělého neuronu popsaný McCullochem a Pittsem [4]

2.4 Knihovna TensorFlow

Jedná se o knihovnu s otevřeným kódem pro aplikace strojového učení, jako jsou neuronové sítě. Byla vyvinuta Google Brain týmem a je používána ve výzkumu i v produkci společností Google [5]. Uvolněna byla pod Apache 2.0 open-source licencí v listopadu roku 2015 [6]. Knihovna TensorFlow umožňuje vývojářům vytvořit „dataflow" (doslova datový tok) strukturovanými grafy, ty se skládají ze série zpracovávajících jednotek, kde každá taková jednotka je reprezentována matematickou funkcí a každé spojení mezi nimi je vícerozměrné datové pole nebo „tensor". TensorFlow poskytuje API s vysokou úrovní abstrakce pro programovací jazyky Python, C, C++, JavaScript a několik dalších. Vlastní funkce knihovny jsou implementovány převážně v jazyce C++. Aplikace můžou běžet v různých prostředích včetně clusterů v cloudu, iOS a Android zařízeních, klasických i grafických procesorech. Společnost Google vytvořila speciální zařízení tzv. Tensor Processing Unit (TPU), jedná se o specifický integrovaný obvod (hardwarový čip) navrhnutý speciálně pro aplikace strojového učení s knihovnou TensorFlow, avšak lze využít i jiné knihovny [7]. Společnost Google využívá TPU jednotky ve svých datacentrech [8].

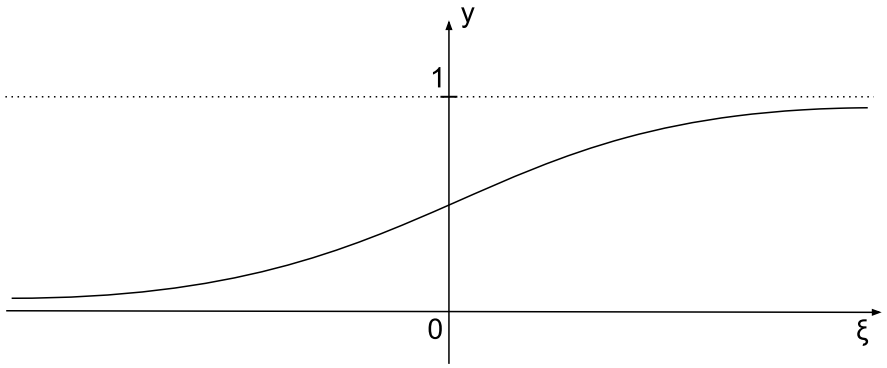
2.5 Učení dopředné neuronové sítě

Učení neuronové sítě je proces, který nastaví síť tak, aby při predikci vracela co nejpřesnější výsledky. V biologických neuronových sítích se samotná struktura sítě od narození téměř nemění, učení je dáno nikoliv přestavbou sítě, ale především posílením nebo naopak zeslabením vazeb prostřednictvím synapsí. Časté a intenzivní impulzy mohou synapse mezi neurony změnit trvale. U umělé neuronové sítě proces učení moderuje pomocí nastavení vah vstupů jednotlivých neuronů, popř. změnou aktivační funkce.



Obrázek 2: Model vícevrstvé neuronové sítě [9]

Model vícevrstvé dopředné neuronové sítě má vstupní vrstvu s n_0 neurony, p skrytých vrstev, každou s n_i neurony, $i = 1, \ldots, p$, a výstupní vrstvu ($p+1$ vrstva) s n_{p+1} neurony, jako na obrázku 2. Výstupní vrstva slouží jen k distribuci vektoru vstupních hodnot $x = (x_1, \ldots, x_{n_0})$ k první skryté vrstvě $x_{n_0} = x_j$, pro $j = 0, \ldots, n_0$. Jednotlivým neuronům je přiřazená přenosová funkce sigmoida (2), zobrazena na obrázku 3.



Obrázek 3: Funkce sigmoida [9]

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}}, \tag{2}$$

kde e je Eulerovo číslo.

Za základ všech učících algoritmů můžeme považovat Hebbův zákon učení, který popisuje změnu synapsí v průběhu učení u živých organismů. Tento předpoklad se pak přenáší i do učení umělých neuronových sítí. Vychází z představy posilování vazeb mezi jednotlivými neurony, které jsou ve stejné chvíli aktivní - excitace. Pokud nedochází k aktivaci ani jednoho z mezi sebou spojených neuronů, vazby mezi nimi naopak slábnou - inhibice. Je-li aktivní pouze jeden neuron z dvojice, synapse zůstává stejná [10].
V umělých neuronech lze toto pravidlo formulovat takto:

$$w_{i+1} = w_i + y_i x_i, \tag{3}$$

kde x_i je vektor vstupních atributů, y_i je informace o zařazení příkladu x_i do třídy, w_i je váha před modifikací, w_{i+1} je váha po modifikaci.

V případě takzvaného učení s učitelem, což můžeme interpretovat lidskýnm učením typu „pokus-omyl", je použita trénovací množina, která se skládá ze vstupů a očekávaných výstupů. Pomocí těchto dvojic jsou pak nastaveny váhy mezi jednotlivými neurony tak, aby se dosáhlo co možná nepřesnějšího požadovaného výsledku. To znamená určit takové váhy, aby se minimalizovala chybová funkce, určená vztahem:

$$E(s) = \sum_{i=1}^{N} [y_i(s,k) - y_i(k)]^2, \tag{4}$$

kde N je počet dvojic trénovací množiny, k je pořadové číslo vzoru v trénovací množině, $y_i(s,k)$ představuje skutečnou hodnotu výstupu sítě na vstupní vektor, $y_i(k)$ je hodnota výstupu sítě na vstupní vektor.

Implementaci učení sítě si můžeme demonstrovat v jazyce JavaScript pro knihovnu TensorFlow na jednoduché funkci XOR. Model má dva vstupní neurony a jeden výstupní. Učící funkce `fit` má jako parametr 30 opakování a celá je spuštěna 100 krát. Po natrénování vypíše odhad blízký konstantě `target_data`.

```
const training_data = tf.tensor2d([[0,0],[0,1],[1,0],[1,1]]);
const target_data = tf.tensor2d([[0],[1],[1],[0]]);

for (let i = 1; i < 100 ; ++i) {
        var h = await model.fit(training_data, target_data, {epochs: 30});
        console.log("Loss after Epoch " + i + " : " + h.history.loss[0]);
}

model.predict(training_data).print();
```

Výpis 1: Ukázka implementace učení neuronové sítě v TensorFlow.js

2.6 Node.js

Jedná se o multiplatformní běhové prostředí s otevřeným kódem a rozsáhlou aktivní komunitu uživatelů. Dokáže spustit JavaScriptový kód mimo prohlížeč. Postavený je na V8 JavaScript interpretu od společnosti Google. Komunita okolo Node.js poskytuje mnoho otevřených knihoven pro práci s umělou inteligenci a zpracováním obrazů.

2.7 Knihovna Jimp

Jedná se o knihovnu určenou pro digitální zpracování a manipulaci s obrazy. Je napsána pouze pomocí JavaScript kódu, a tedy nemá žádné další závislosti což umožňuje snadné ovládání z čehož

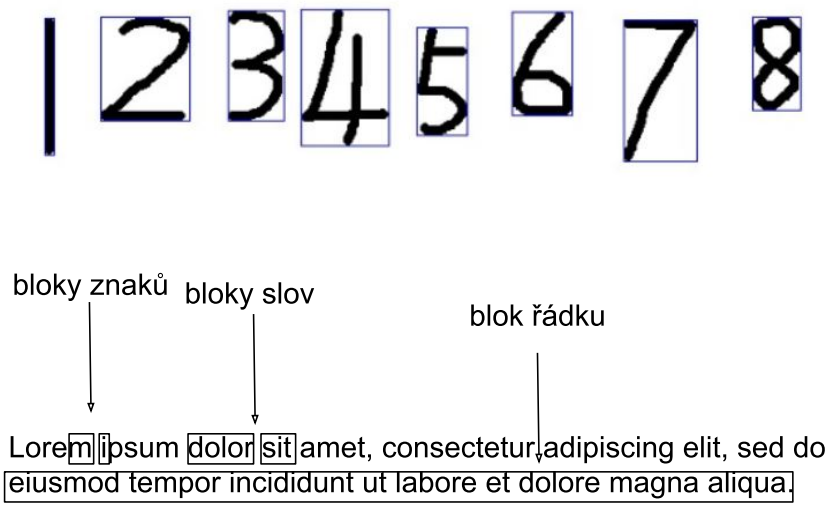
na druhou stranu vyplývají určité omezení, při využití ve vícevláknových aplikacích. Poskytuje algoritmy pro konverzi do různých formátů, úprava kontrastu, odstranění barev, funkce pro zmenšení, zvětšení, škálování a vyřezávání obrazů. V současné době má přes 500 000 stažení týdně [8]. Podporuje formáty typu JPEG, PNG, BMP, TIFF a GIF. Nevýhodou je sekvenční přístup bez využití grafické jednotky.

2.8 OCR neboli optické rozpoznávání znaků

Je konverze obrazů psaného, rukopisného nebo tištěného textu do strojově kódovaného textu, ať už z naskenovaného dokumentu, fotografie či textu na dopravních značkách. Může být použito například pro zpracování různých dokumentů (jako například šek, faktura nebo, bankovní výpis), automatické rozpoznávání poznávacích značek aut, pro kontrolu pasů na letištích, skenování knih, převod rukopisu do digitální podoby v reálném čase a další.

2.9 Tesseract OCR Engine

Tesseract je OCR (Optické rozpoznávání znaků z anglického Optical Character Recognition) engine s otevřeným kódem, který vyvinula společnost HP mezi lety 1984 a 1994. Od roku 2006 je vyvíjen a udržován společností Google. Dokáže rozpoznat více než 100 jazyků. V první fázi procesu rozpoznávání se shromažďují obrysy znaků do jednotlivých bloků, naznačeno na obrázku 4, ty jsou uspořádány do řádků, poté jsou rozděleny do slov a řádků podle mezer mezi znaky, takové mezery jsou snadno rozpoznány pomocí jednotlivých buněk. Ve druhé fázi je při prvním průchodu proveden pokus rozpoznat každé slovo. Vyhovující je předáno adaptivnímu klasifikátoru jako tréninková data. Vzhledem k tomu, že adaptivní klasifikátor se může naučit něco užitečného projde stránku druhým průchodem, ve kterém rozpozná i slova, která nebyla s jistotou rozpoznána již při prvním průchodu. Závěrečná fáze řeší mezery a kontroluje alternativní možnosti tam, kde v předchozích fázích nebyl text určen se spolehlivou pravděpodobností [11].



Obrázek 4: Ukázka vytvořených obrysů znaků

2.10 Podobnost textových řetězců - Diceho Koeficient

Jedná se o algoritmus vyvinutý Thorvald Sørensen a Lee Raymond Dicev roce 1948 [12]. Koeficient měří podobnost mezi množinami X a Y (5). Pokud jsou identické (tj. obsahují stejné prvky), koeficient se rovná 1 zatímco pokud X a Y nemají žádné společné prvky je roven 0. V případě částečné shody mezi 0 a 1. Výpočet podle vzorce (5). Jako implementaci v Node.js použiji `compareTwoStrings` funkci z knihovny string-similarity [13].

$$\frac{2(|X| \cap |Y|)}{|X| + |Y|}, \tag{5}$$

kde X, Y je počet prvků, $|X|$ je mohutnost množiny X, $|Y|$ je mohutnost množiny Y.

```
stringSimilarity.compareTwoStrings('Zkouska', 'Zkooska'); // výsledek: 0,85
```

Výpis 2: Ukázka implementace Diceho Koeficientu v JS

3 Analýza

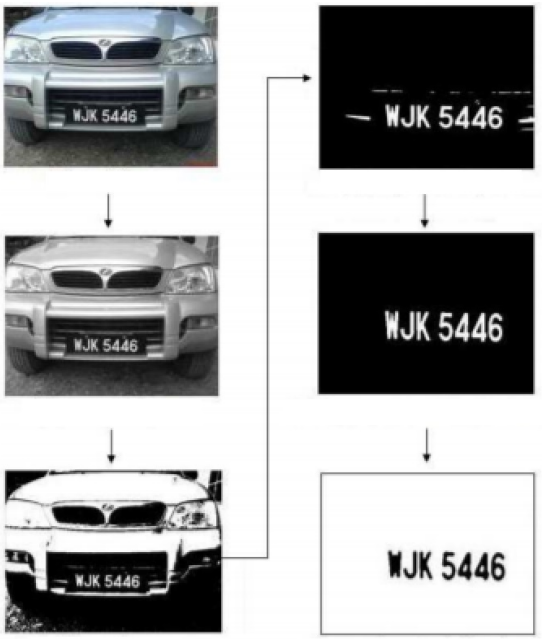
V předchozí kapitole byly představeny základní pojmy a knihovny. Tato kapitola bude sloužit k rozboru podobných problémů, již řešených v minulosti, souvisejících s detekcí textu na objektech v obrazu. Kromě toho také k nastínění celkové problematiky řešeného problému, tedy nalezení a přečtení cedule s názvem ulice. Dále se bude věnovat analýze a popisu mého řešení.

3.1 Řešení problémů podobných rozpoznávání ulic z obrazu

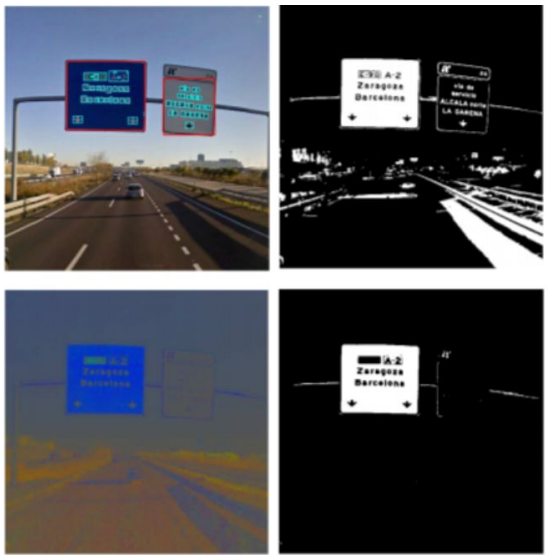
K podobným problémům jako je rozpoznávání názvů ulic z obrazu patří například rozpoznání registrační značky vozidla například L.Angeline K.T.K., Teo, Farrah Wong používají tzv. smearing algoritmus [14]. Po načtení obrázku dojde k odstranění přebytečných barev převodem do černobílé podoby, poté do binární podoby, dále k odstranění oblastí bílých pixelů dotýkajících se hranice obrazu a poté k odstranění šumu, zvětšení a předání do části OCR obrázek 5. Další podobný problém je například rozpoznání textu na dopravních panelech řešený v práci A. González, L.M. Bergasa, J. Javier Yebes and J.Almazán [15]. Princip detekce cedule je zde podobný předchozí práci [14], postup naznačen na obrázku 6 a pro OCR využívají Hidden Markov Models (HMMs). Rozpoznaný text poté využívají ke geolokačním účelům. Při experimentech ve vzdálenosti menší než 30 metrů výsledky vypadají následovně.

Tabulka 1: Publikované výsledky rozpoznávání na krátkou vzdálenost

Data	Úspěšnost detekce	Úspěšnost rozpoznán
Slova	92,00%	67,21%
Čísla	73,35%	64,13%
Symboly	63,08%	80,00%

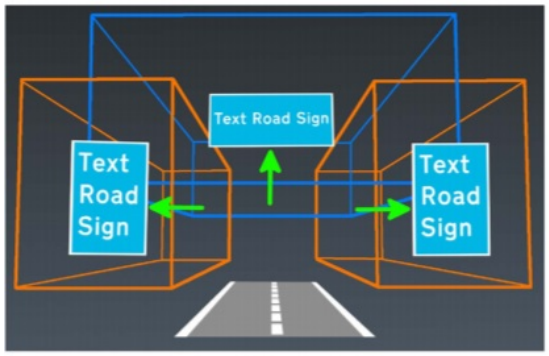


Obrázek 5: Popis smearing algoritmu [14]



Obrázek 6: Ukázka detekce a segmentace dopravních panelů [15]

Další práce na téma rozpoznávání textu pochází od Jack Greenhalgh, Majid Mirmehdi [16]. Jedná se opět o detekci textu z tabulí na vysokorychlostních silnicích a dálnicích. Postup detekce spočívá zaprvé v definování regionů, kde se nachází tabule s využitím detekce směru bílých čar na silnici, ukázáno na obrázku 7. Dále jsou detekovány místa s pravděpodobným výskytem tabulí, obarvené zóny na obrázku 8.

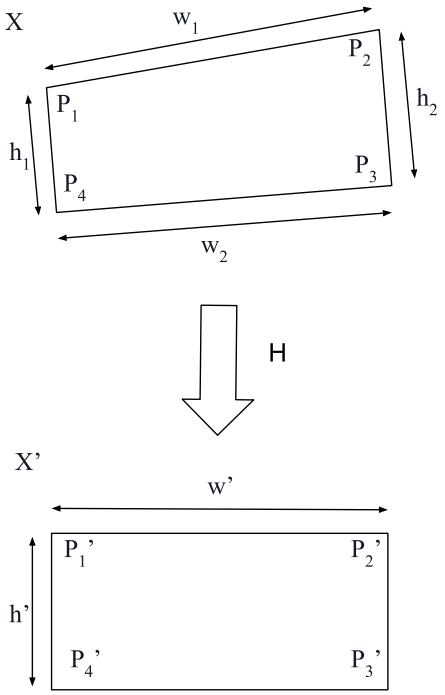


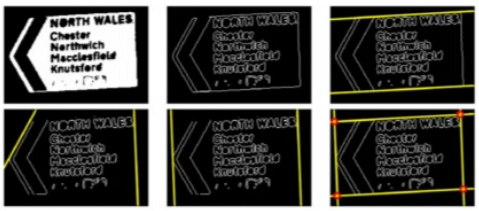
Obrázek 7: Projekce zón do 2D prostoru [16]

Obrázek 8: Obarvení zón s možným výskytem tabulí [16]

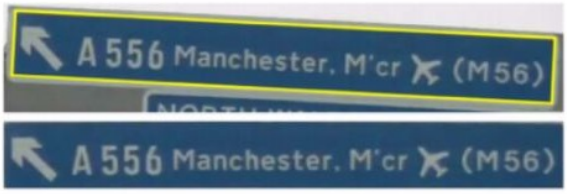
Po binarizace regionu jsou identifikovány rohy. Z nich je poté sestaven čtverec, který udává sklon tabule, zobrazeno na obrázku 9. Tabule je poté výpočtem a škálovací funkcí (6) srovnána, naznačeno na obrázku 10.

$$w = max\{|P1 - P2|, |P4 - P3|\} h = max\{|P1 - P4|, |P2 - P3|\} \quad (6)$$





Obrázek 9: Ilustrace detekce sklonu tabule [16]



Obrázek 10: Ilustrace škálování tabule [16]

Tabulka 2: Publikované výsledky porovnávají s výsledky z předchozí práce [15]

Metoda	Koeficient přesnosti (metodologie [16])
Jejich metoda	0,93
González et al.[15]	0,60

3.2 Typy cedulí s názvy ulic

Typů a barevných provedení cedulí nesoucí název ulice existuje v České republice několik, obvykle se využívají státní barvy (bílá, červená, modrá) např. rada hlavního města Prahy usnesením č. 1267 z 28. 8. 2012 schválila jednotný vzhled uličních tabulí písmo „SMALT" a barvy (červená RAL 3000, bílá RAL 9003), ukážka na obrázku 11.

Obrázek 11: Příklady cedulí s názvem ulice *(www.google.com/maps)*

Dále se rozpoznávací algoritmus musí vypořádat s problémy jako jsou různě nakloněné, deformované a špatně čitelné značky, s překážkami ve výhledu či různými škrábanci, nálepkami a vandalismem nebo více cedulí na jednom snímku, jako na obrázku 12.

Obrázek 12: Příklady problematických cedulí *(www.google.com/maps)*

3.3 Dostupné metody řešení problému

Podobný problém byl již řešen společností Google například v práci Raymonda Smitha a dalších [17]. Jejich model je založen na (deep recurrent neural network) hluboké rekurentní neuronové síti, která se naučí, jak identifikovat názvy ulic (ve Francii) z obrázku obsahujícího až čtyři různé pohledy na název ulice obrázek 13. Model slučuje informace z různých pohledů a normalizuje text do správného formátu.

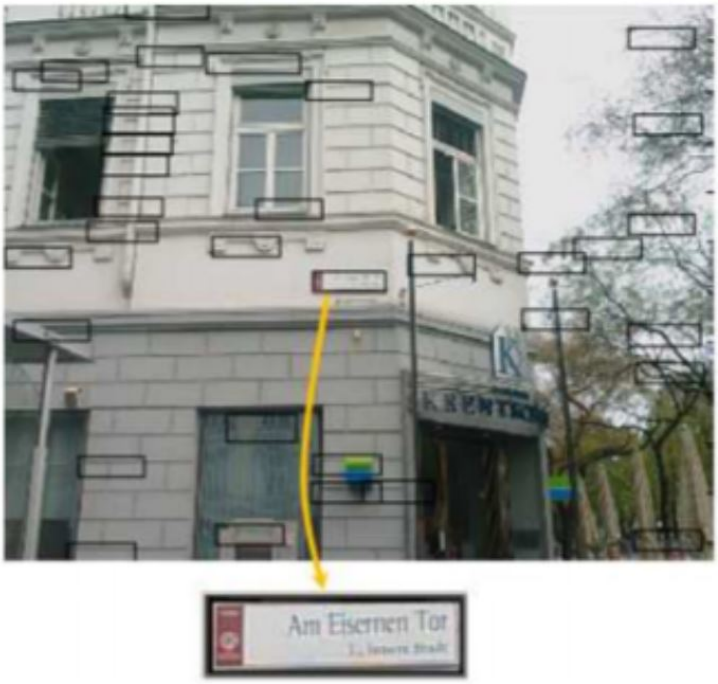
Obrázek 13: Příklady cedulí s názvem ulice z FSNS datasetu [17]

Se 40 paralelními stroji ve clusteru a po 30 milionech učících cyklech chybovost detekce dosahuje přibližně 25%. Během svého výzkumu vytvořili FSNS dataset obrázků o velikosti více než 158 Gb.

Stejným problémem se zabývali Sobhan Parizi a Alireza Tavakoli v práci [18]. Model neuronové sítě, který používají je naučen z 100 pozitivních and 7000 negativních příkladů, ukázka na obrázku 14. Na vstupu je 20×70 pixelů velký obrázek s hodnotou pixelu normalizovanou v intervalu [0, 1] a výstupem jsou souřadnice 4 bodů. Po aplikaci vícero náklonů a škálování na vstupní obrázek obdrží několik falešně pozitivních výsledků, označené zóny na obrázku 16 a pomocí další neuronové sítě oddělí vlastní text ze značky od pozadí, naznačeno na obrázku 15. S využitím 5000 slov velké Anglické databáze dosahovala úspěšnost na 1000 vzorcích s textem v dobré kvalitě 96,51% [18].

Obrázek 14: Ukázka datasetu z práce [18]

Obrázek 15: Ukázka detekce textu na ceduli [18]



Obrázek 16: Ukázka falešne pozitivních výsledků [18]

3.4 Vlastní návrh řešení problému

Tato podkapitola je věnována mému postupu při návrhu řešení problému. Ukázkou a popisem způsobu nalezení cedule, její následné zpracování, rozpoznání a případné zpřesnění výsledku.

3.4.1 Postup

Pokusil jsem se navrhnout postup, který by dokázal rozpoznat název ulice i z obrázků ve velkém rozlišení. Problém jsem rozdělil na dvě hlavní části:

1. Nalezení tabulky s názvem ulice,
2. rozpoznání textu.

3.4.2 Návrh modelu neuronové sítě a použitý dataset

K nalezení tabulky s názvem ulice využijí neuronovou síť s 2304 vstupními neurony jednou skrytou vrstvou a jedním výstupem. Velikost obrázků v datasetu je $32 \times 18 \times 4$ pixelů (šířka, výška, RGBA). Výstup udává zda-li daný vstupní obrázek je cedulí popisující název ulice nebo není. Příklad dat pro učení neuronové sítě:

Obrázek 17: Příklady cedulí s názvem ulice pro hodnotu 1 *(www.google.com/maps)*

Obrázek 18: Příklady cedulí s názvem ulice pro hodnotu 0 *(www.google.com/maps)*

Jednou z důležitých vlastností datasetu pro hodnotu 1 obrázek 17 je, aby se na obrázku zobrazovala cedule, pokud možno celá včetně okrajů, tato vlastnost je zásadní pro správné nalezení cedule v algoritmu prohledávání obrazu. Sestavený dataset obsahuje 100 pozitivních obrázků a 5000 vzorků pro negativní výsledek. Pro sestavení negativních příkladu obrázek 18 byly využity obrázky, které neuronová síť v předchozích měřeních označila jako „falešně pozitivní", pro tento přístup jsem se inspiroval v práci Sobhan Parizi a Alireza Tavakoli [18]. Do modelu jsou přidány 2 skryté vrstvy s 2304 neurony na vstupu a výstupní vrstva s jedním neuronem.

```
var numberOfNeurons = 2304;
var model = tf.sequential();

const hidden = tf.layers.dense({
        units: 2,
        inputShape: [numberOfNeurons],
        activation: 'sigmoid'
});
model.add(hidden);

const output = tf.layers.dense({
        units:1,
        activation: 'sigmoid'
});
model.add(output);
```

Výpis 3: Implementace definice neuronové sítě v Tensorflow.js

3.4.3 Algoritmus pro prohledání obrázku

Samotný algoritmus má za úkol vstupní obrázek projít a každou jeho část předat v normalizovaném tvaru tedy o velikosti 32×18 pixelů neuronové síti která danou část označí pravděpodobností, zda se jedná o tabulku s názvem ulice nebo ne. Postup algoritmu bude nejlepší demonstrovat na sledu obrázků v první fázi probíhá „hrubé" prohledání: Algoritmus je rozdělen do dvou fází, v první je provedeno „hrubé" rozdělení obrázku. Kolem obrázku je vytvořen pomyslný obdelník a v prvním průchodu cyklem je obrázek zmenšený do normalizovaného tvaru 32×18 a ohodnocen, výsledky z jednotlivých obdelníků naznačeny na obrázku 19. V dalším průchodu je obdelník zmenšený vertikálně (pro ilustraci víc, než je použito v implementaci) a z obdelníku je opět vytvořen formalizovaný tvar a ohodnocen, jako na obrázku 20. V následujících krocích vnořeného cyklu algoritmu je obdelník posouván směrem doprava dokud "nenarazí" na stěnu, při každém posunu je opět ohodnocen neuronovou sítí. Po ukončení vnořeného cyklu je obdelník zmenšený i horizontálně, ilustrovaném na obrázek 21. A následně je posouván po obrazu v pořadí (červená, zelená, modrá, žlutá), přičemž pokaždé je ohodnocen neuronovou sítí, jako na obrázku 22.

Obrázek 19: Ukázka oblasti pro zmenšení v první iterace

Obrázek 20: Ukázka oblasti pro zmenšení ve druhé iteraci

Obrázek 21: Ukázka oblasti ve třetí iteraci

Obrázek 22: Znázornění posunu obdelníku po obrazu

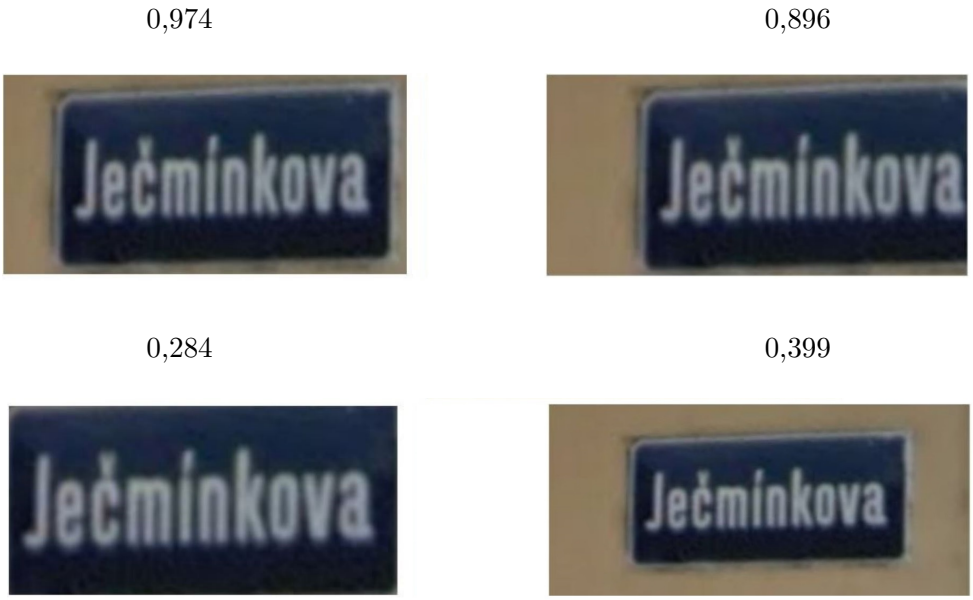
Algoritmus končí, když je velikost obdelníku příliš malá na to aby se v něm mohla nacházet čitelná cedule (v mé implementaci je použita velikost 64×36 pixelů). Po průchodu algoritmu celým obrázkem je výsledkem to že obdelníky s „lepším" hodnocením od neuronové sítě se shlukují kolem cedule, naznačeno na obrázku 23, ale obdelníky, které jsou již uvnitř cedule nemají tak dobré hodnocení, protože v testovacích datech je obsažen i obdelník cedulí.

Obrázek 23: Ukázka výsledných obdelníků s vysokým hodnocením

Obdelník, který byl ohodnocen nejlépe obrázek 24 je převzat a v závislosti na své velikosti je původní obraz vyřezán. Na takto vytvořený nový obrázek je aplikován celý algoritmus znovu, ale nyní je průchod mnohem „jemnější" (obdelník je posouván a zmenšován po menších částech) a ukončení algoritmu je možné provést již po zmenšení na velikost polovinu původního obdelníku, po opakované aplikaci algoritmu obdelník s nejlepší hodnotou vypadá v ideálním případě následovně obrázek 25. Příklad hodnocení obrázků natrénovanou neuronovou sítí obrázek 26 - vyšší číslo znamená vyšší pravděpodobnost, že se jedná o tabulku s názvem ulice.

Obrázek 24: Ukázka nejlepšího nalezeného pohledu při prvním průchodu algoritmem

Obrázek 25: Ukázka nalezené oblasti při druhém, přesnějším průchodu



Obrázek 26: Příklad hodnocení pohledů na značku neuronovou sítí

Ukázka zjednodušené verze algoritmu:

```
//cyklus pro zmenšování obdelníku
while (frameWidth > stopWidth && frameHeight > stopHeight) {
        //podmínka rozhodující zda bude obdelník zmenšený vertikálně nebo
            horizontálně
        if (resizeWidth) {
                frameWidth = frameWidth * percentilReisze;
        }
        else {
                frameHeight = frameHeight * percentilReisze;
                resizeWidth = !resizeWidth;
        }
        //cyklus pro posun obdelníku vertikálně
        for (var a = frameX; (a + frameWidth) <= image.bitmap.width; a = a +
            posunX) {
```

```
                //cyklus pro posun obdelníku horizontálně
                for (var b = frameY; (b + frameHeight) <= image.bitmap.height; b
                     = b + posunY)

                    //funkce pro predikci obrazu neuronovou sítí
                        var predict1 = predict(image.clone().crop(a, b,
                            frameWidth, frameHeight).resize(32, 18))

                if (predict1 > predictMax) {
                        /*hodnoty obdelníku s největší hodnotou jsou uloženy zde,
                             pro další průchod */
                }
            }
        }
    }
}
```

Výpis 4: Zjednodušená implementace algoritmu

3.4.4 Předzpracování nalezeného obrazu pro tesseract

Jako první je aplikována funkce `greyscale()` a `invert()` (podle původních barev na obrázku 27 tak, aby text byl černý na bílém, ukázka na obrázku 28). Poté je zmenšený o 10 % svrchu a zespodu, naznačeno na obrázku 29. Dále je obraz binarizován, tedy všechny tmavé pixely jsou černé a všechny světlejší pixely, než práh budou bílé, ukázka na obrázku 30. Poté je analyzován pomocí OCR.

Obrázek 27: Příklad nalezené cedule

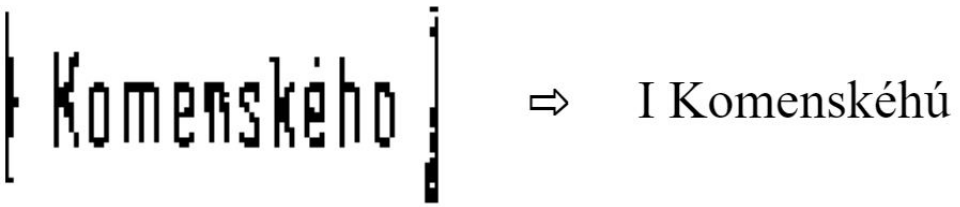
Obrázek 29: Ukázka černobílé cedule po zmenšení

Obrázek 28: Příklad černobílé cedule

Obrázek 30: Ukázka cedule po binarizování

3.4.5 OCR

K rozpoznání textu je použitý program Tesseract.js s nastavením českého jazyka. Výstupem například z následujících obrázků obrázek 31 a obrázek 32, je tento řetězec.



Obrázek 31: Příklad rozpoznání textu z cedule

Obrázek 32: Příklad rozpoznání textu z cedule

3.4.6 Zpřesnění pro české názvy ulic

Jednou z možností, jak vylepšit, či zpřesnit výsledky je využit funkci, kterou poskytuje knihovna Tesseract `user_words_file`, tedy použít textový dokument, který bude obstahovat názvy ulic v České republice. Takový seznam je možné získat buď na oficiálních webových stránkách jednotlivých měst, případně jej extrahovat ze služeb jako například OpenStreetMap. Po aplikování vytvořeného slovníkového souboru s ulicemi v Praze tesseract správně rozpoznal výše uvedené slovo, tedy „Komenského".

4 Experimentální část

Tato kapitola slouží především k prezentování dosažených výsledků během testování dostupného modelu a vlastního řešení. Dále pak k seznámení s použitým vybavením, uvedením nedostatků řešení a návrhů k případných vylepšení.

4.1 Dostupné modely a jejich výsledky

Povedl jsem testování na části mého datasetu (30 obrázků ve vysokém rozlišení) s českými popisky ulic na programu s otevřeným kódem , který vytvořili Jack Greenhalgh a Majid Mirmehdi v práci [17] implementovaném v jazyce Python. Pro tento experiment bylo třeba můj dataset upravit do přijatelné formy (4 pohledy). K učení modelu jsem použil jejich volně dostupný dataset a sestavu Intel Core i7-5500U CPU 2,4 GHz, 16 GB RAM, po přibližně 24 hodinách učení se chybovost blížila hranici 35%.

Při měření na mém datasetu cedulích v Praze model nebyl schopen rozpoznat obstojně text ani z jednoho obrazu. Výstupem byly pouze náhodné znaky. Se zmenšováním vstupních obrazů (přiblížení pohledu k ceduli) se úspěšnost zvýšila. Správně rozpoznán sice nebyl ani jeden text, ale již zde byly náznaky, např. výše zmíněnou tabulku „Komenského" model rozpoznal jako „Kumenskghu". Model tedy není vhodný k rozpoznání malých cedulí na velkých obrazech.

4.2 Výsledky vlastního postupu

V předchozí kapitole byly prezentováný výsledky testování vlastního datasetu na již dostupném modelu a tato podkapitola bude zaměřena na dosažené výsledky vlastního řešení.

4.2.1 Použitý hardware

1. Intel Core i7-5500U CPU 2,4 GHz, 16 GB RAM (dále jen osobní PC)
2. Na stroji poskytnutým Katedrou informatiky VŠB-TU Ostrava merlin1.cs.vsb.cz [19] s procesorem Intel Xeon CPU E5-2640 v2 @ 2.00GHz (dále jen merlin1)

4.2.2 Učení neuronové sítě

Učení neuronové sítě na 100 pozitivních a 5000 negativních značkách trvalo na stroji merlin1 zabralo 2 hodiny a po 10 000 cyklech (při každém 100 opakování) učení bylo dosaženo odchylky menší než 0,0174.

4.2.3 Rychlost analýzy značky

Proces nalezení značky je na celém řešení nejnáročnější na hardwarový výkon. Průměrná doba analýzy obrazu na stroji Intel Core i7-5500U CPU 2,4 GHz, 16 GB RAM je 2,3 minuty a analýza jednoho obrazu zabrala v průměru 42 sekund na vzorku 100 obrazů.

4.2.4 Výsledky testování na vzorku 100 obrázků s velkým rozlišením

Celý test jsem provedl dvakrát, první bez slovníku s českými názvy ulic a pote s ním. Jeden test trval 70 minut. K porovnání rozpoznaného řetězce s správným jsem použil řešení založené na tzv. Diceho Koeficientu [13] [12].

Tabulka 3: Výsledek měření bez slovníku s českými názvy ulic

Počet testovacích obrazů	Počet obrázků se 100% shodou	Průměrná hodnota Diceho Koeficientu
100	13	0,656

Tabulka 4: Výsledek měření s aplikací slovníku

Počet testovacích obrazů	Počet obrázků se 100% shodou	Průměrná hodnota Diceho Koeficientu
100	81	0,856

Z měření pozorujeme, že při aplikaci pomocného slovníku dokázal Tesseract rozpoznat podstatně více textu. Obrazy, které byly chybně rozpoznány v druhém testu měli méně kvalitní a malou ceduli s nekvalitním a hůře čitelným textem. Dobře čitelné cedule byly rozpoznány správně.

4.2.5 Možná vylepšení paralelizací

Jednou z hlavních problémů je sekvenční implementace algoritmu pro vyhledání v obrazu, i přes to že samotná predikce z obrazu probíhá paralelně pomocí TensorFlow jádra, použitá knihovna pro práci s grafikou, Jimp paralelní implementaci neumožňuje a zpracování obrazů probíhá pomocí CPU, tato část tedy zabírá nejvíce strojového času. Při použití jiné metody pro digitální zpracování obrazu s pomocí GPU lze očekávat řádové zrychlení celé aplikace [20].

5 Závěr

V teoretické části bakalářské práce jsem uvedl stručný přehled technologií, které jsou používané v oblasti strojového učení, rozpoznávání a zpracování obrazů. Prozkoumal jsem a otestoval dostupné modely řešící podobné problémy. S pomocí získaných znalostí jsem se pokusil sestavit vlastní algoritmus schopný nalézt a přečíst ceduli. Nevýhodou mé implementace je zpracovávání obrazu pomocí CPU, která je hlavní příčinou velké časové náročnosti samotného rozpoznávacího procesu. Jako návrh budoucího rozšíření práce je přepracování algoritmu do paralelní implementace, kde se dá očekávat řádové zrychlení celého procesu nalezení tabule s názvem ulice v obrazu. Při použití slovníku českých názvů ulic při OCR dojde ke značnému zpřesnění rozpoznávání, dala by se však vylepšit i metoda přípravy obrazů pro samotné rozpoznávání použitím další neuronové sítě, která by identifikovala přesné umístění textu v již nalezené ceduli a pravděpodobně by došlo ke značnému zpřesnění [18]. Použité zdrojové kódy, datasety a výpisy z měření jsou umístěny v příloze.

Odkazy

1. SOLOMON, Chris; BRECKON, Toby. *Fundamentals of Digital Image*. Wiley; 1 edition (January 4, 2011), 2010. ISBN 0470844736. Dostupné také z: http://www.oalib.com/references/8289819.
2. HUANG, Thomas. *Computer Vision : Evolution And Promise*. 1996. ISBN 978-9290830955.
3. MACKWORTH, Poole; GOEBEL. *Computational Intelligence: A Logical Approach*. 1998. ISBN 9780195685725.
4. MARŠÁLOVÁ, Kateřina. Vlnková transformace jako nástroj pro výběr příznaků k rozpoznávání obrazů. 2017. Dostupné také z: https://is.muni.cz/th/ymg0n/BP-Katerina_Marsalova.pdf.
5. DEAN, Jeffrey. *It is machine learning software being used for various kinds of perceptual and language understanding tasks*. 2015. Dostupné také z: https://www.youtube.com/watch?v=oZikw5k_2FM.
6. *Tensorflow web*. Dostupné také z: https://www.tensorflow.org/about.
7. SHETH, Rajen. Introducing PyTorch across Google Cloud. 2018. Dostupné také z: https://cloud.google.com/blog/products/ai-machine-learning/introducing-pytorch-across-google-cloud.
8. MORAN, Oliver. *Jimp web*. Dostupné také z: https://www.npmjs.com/package/jimp.
9. BISKUP, Roman. Možnosti neuronových sítí. 2009. Dostupné také z: https://www.pef.czu.cz/dl/46225.
10. ZÁLESKÁ, Kateřina. Klasifikace obrazů pomocí umělých neuronových sítí. 2011. Dostupné také z: https://is.muni.cz/th/mqtzd/bakalarska_prace.pdf.
11. SMITH, Ray. An Overview of the Tesseract OCR Engine. 2007. Dostupné také z: https://static.googleusercontent.com/media/research.google.com/cs//pubs/archive/33418.pdf.
12. CHEN, Hao. String Metrics and Word Similarity applied to Information Retrieval. 2012. Dostupné také z: https://pdfs.semanticscholar.org/326b/35240206437e903f6b4a28e4a8bc03670pdf.
13. KURDEKAR, Akash. *Node.js package pro výpočet podobnosti stringu používající tzv. Sørensen–Dice coefficient*. Dostupné také z: https://www.npmjs.com/package/string-similarity.
14. L.ANGELINE; TEO, K.T.K.; WONG, Farrah. *Smearing Algorithm for Vehicle Parking Management System*. Wiley; 1 edition (January 4, 2011), 2009. ISBN 978-967-5224-20-1. Dostupné také z: https://msclab.files.wordpress.com/2011/01/smearing-algorithm-for-vehicle-parking-management-system.pdf.

15. GONZÁLEZ, A.; BERGASA, L.M.; YEBES, J. Javier; J.ALMAZÁN. *Text Recognition on Traffic Panels from Street-level Imagery*. 2012. ISBN 978-1-4673-2118-1. Dostupné také z: http://www.robesafe.com/personal/javier.yebes/docs/Gonzalez12iv.pdf.

16. GREENHALGH, Jack; MIRMEHDI, Majid. Recognizing Text-Based Traffic Signs. 2015. Dostupné také z: https://www.researchgate.net/publication/276482204_Recognizing_Text-Based_Traffic_Signs.

17. SMITH, Raymond; GU, Chunhui; LEE, Dar-Shyang; HU, Huiyi; UNNIKRISHNAN, Ranjith; IBARZ, Julian; ARNOUD, Sacha; LIN, Sophia. End-to-End Interpretation of the French Street Name Signs Dataset. 2017. Dostupné také z: https://link.springer.com/content/pdf/10.1007/978-3-319-46604-0_30.pdf.

18. PARIZI, Sobhan Naderi; TARGHI, Alireza Tavakoli; AGHAZADEH, Omid; EKLUNDH, Jan-Olof. *Reading Street Signs using a Generic Structured Object Detection and Signature Recognition Approach*. 2009. ISBN 978-989-8111-69-2. Dostupné také z: http://www.csc.kth.se/~sobhannp/docs/papers/VISAPP09_UrbanPlaceRecognition.pdf.

19. *Stroje pro práci se strojovým učením Katedra informatiky FEI VŠB-TU Ostrava*. Dostupné také z: http://wiki.cs.vsb.cz/index.php/Merlin.

20. LADKAT, Ajay S.; DATE, Abhijit A.; INAMDAR, Suyash S. *Development and comparison of serial and parallel image processing algorithms*. 2016. ISBN 978-1-5090-1285-5. Dostupné také z: https://ieeexplore.ieee.org/document/7824894.

Adresářová struktura přílohy

1. /bc bakalářská práce ve formátu pdf
2. /model složka s uloženým modelem ve formátu JSON
3. /trainingData složka s datasetem pro učení
4. /images složka pro testování obrazů modelem